Amendments to the Specification

Please amend the Abstract as follows, where the double brackets indicate deletions:

The invention proposes an encoding and a decoding method to be used for transmitting and storing description element(s) of an XML-like document ~~which is an instance of an XML-like schema~~. ~~The methodology includes: using at~~ At least one table ~~derived from said schema, said table~~ containing identification information is used for ~~solely~~ identifying each description element in a hierarchical level, and structural information for retrieving any child description element from its parent description element[[,]]; ~~scanning~~ a hierarchical memory representation of the instance from parent description elements to child description elements is scanned until reaching the description element to be encoded[[,]]; ~~and retrieving~~ the identification information of each scanned description element[[,]] is retrieved; and ~~encoding~~ said description element ~~to be~~ is encoded as a fragment ~~comprising said~~ that includes content and a sequence of the retrieved identification information.

Please amend page 10, line 33 to page 11, line 8 as follows:

A method for decoding a fragment will now be described by reference to figure 3. According to figure 3, a decoding method according to the invention consists in:

- step 3-1: finding in the table the description element (DE) associated to the received sequence of identification information,

- step 3-2: decoding (BiM-D) the received content (C(DE)) according to the primitive type of said description element (found in the table),
- step 3-3: updating the hierarchical memory representation (DM-D) by adding said element together with its content; adding its parent description element if they are missing; and in case of multiple occurrences, adding same description elements of lower rank if they are missing.

Please amend page 20, line 15 to page 21, line 6 as follows:

Figure 5 gives an example of a binary encoding for the compact key <<0.1[70][1]>>. Five bytes are needed to encode the compact key <<0.1[70][1]>>. Each byte starts with two control bits (CB). The six less significant bits are used to encode the value. The control bits of the first byte are '00' (new level). Its value bits (VB) are '000000' which is the binary representation of the first identification information of the sequence ('0'). The control bits (CB) of the second byte are '10' ('indexed'). Its value bits (VB) are '00001' which is the binary representation of the second identification information of the sequence ('1'). The binary representation of the first index '70' is '1000110' which contains more than six bits. Therefore the encoding is done on two bytes: the third and the fourth bytes. The control bits (CB) of the third byte are '01' (continue). Its value bits (VB) are '000110' (less significant bits of the index to be encoded). The control bits (CB) of the fourth byte are '10' (indexed). Its value bits (VB) are '000001' (most significant bits of the index to be encoded). Finally the control bits (CB) of the fifth byte are '11' (end). And its value bits (VB) are '000001' (binary value of the index to be encoded).

Please amend page 21, lines 7-10 as follows:

Figure 6 gives an example of a binary encoding of the data size 575 (binary: 10 00111111). The first byte is composed of the 7 less significant bits of the length value (VB) with the addition of a control bit (CB) specifying that another byte is required. The second byte contains the remaining bits (VB) with the "end" control bit (CB).